SYSTEM AND METHOD FOR EXCHANGING DATA AND COMMANDS BETWEEN AN OBJECT ORIENTED SYSTEM AND RELATIONAL SYSTEM

This application contains a Microfiche Appendix consisting of one (1) slide and 30 frames.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to systems and methods for transferring data and commands between computing systems. In particular, the present invention relates to a system and a method for exchanging data and commands between an object oriented system and a relational system.

2. Description of the Background Art

With the development and proliferation of computers of increasing performance capability, a number of different languages and programming paradigms have been developed. These languages and programming paradigms are used in conjunction with data that can be stored persistently in a variety of different ways. For example, options for persistent storage include relational databases, file systems and object-oriented databases.

Most data has been stored in a relational format such as in tables of a relational database. The data is manipulated and maintained by a relational database management system (RDBMS). One particularly attractive attribute of such relational systems is that RDBMSs are able to persistently store data. Relational systems are persistent in the sense that the data is stored in a stable storage environment such that the data is accessible even after the application that created the original data stops executing. Furthermore, there are a number of applications and tools available for the manipulation and maintenance of such data in relational databases. Since relational databases have been in existence for many years, the use and proliferation of such applications and tools are widespread, and sophistication and capabilities of the tools are great.

However, a new programming paradigm that has become more widespread in recent years is object-oriented programming (OOP). In fact, OOP is becoming the dominant programming paradigm with the development and widespread use of new programming languages such as JAVA and C++. In OOP, a class is used to encapsulate the structure and behavior of objects. Thus, the objects contain both the data and the functionality for manipulation of the data.

It is very natural and desirable for application developers to represent the business objects in an object-oriented language like Java and at the same time use RDBMS for the persistence storage of those objects.

One problem existing in the art is that there are no systems and methods to bridge the gap between the programming paradigm used for object-oriented systems and the programming paradigm used for relational systems. There are also no systems and methods for bridging the gap between the languages used for object-oriented systems such as Java and the languages used in relational systems such as SQL. Furthermore, there is no easy method for specifying the mapping between such object-oriented systems and relational systems. Thus, to perform translation between these systems has required hand coding of the mapping between object-oriented systems and relational systems, and hand coding is tedious, time-consuming, error-prone and tends to be non-uniform. Therefore, there is a need for systems and

method for automatically translating and exchanging data between object-oriented systems and relational systems.

The prior art has attempted to solve the problem with graphical user interfaces that define mappings and by producing proprietary platform specific code that will translate between object-oriented systems and relational systems. However, such prior art systems have the following shortcomings. First, they are not always able to create a relational schema given an object model. Second, they are not always able to produce an object model given a relational schema. Third, they do not provide a uniform method for specifying directed options for object graph specification for different operations. Fourth, the prior art is not able to handle large sets of queried objects by streaming them between different tiers of applications. Thus, they are subject to memory bandwidth and response-time performance problems. Finally, such existing systems are coded for operation with a particular RDBMS. Thus, they are not interoperable among different relational back ends.

In object-oriented systems, when a new object is created it is typically assigned an identification number that can identify it uniquely among other objects of the same type. If objects are stored persistently in a database, the identification number assigned to a newly created object in memory should be unique with respect to even the already stored objects in the database. So there is a need for a system and method having the ability to always provide a unique number. The existing art has attempted to solve the problem of providing unique identification numbers by providing the notion of a unique row-id in the RDBMS, thereby assigning a newly inserted row a unique number. However, the prior art approach has a number of disadvantages. First, the unique id is not known to the program until the object is inserted in the database. So if the programmer has to create related objects which need to know the unique ids for their initialization, the current scheme would require an insert operation and then a query operation to get the RDBMS assigned unique id. This is inefficient and cumbersome. Second, not all RDBMSs have the feature of unique row-ids, thus, such systems in the prior art cannot generate unique ids. Third, each RDBMS specifies its own unique way of defining and retrieving these row ids. So the application programmer cannot use a consistent and portable way of defining, using and referring to the unique ids.

Therefore, because of the advantages offered by OOP and the persistence of data offered by relational systems there is a need for a system that can easily be configured and that can reliably and automatically transfer data between such relational systems and object-oriented systems.

SUMMARY OF THE INVENTION

The present invention overcomes the deficiencies and limitations of the prior art with a system and methods for exchanging data and commands between an object oriented system and a relational system. In particular, the system of the present invention comprises an Object-Relational Mapping (ORM) grammar, an ORM specification, Object Class Definitions, a relational database, an operating system, a Database Exchange Unit including an OR mapping unit, a schema generator, a schema reverse engineering unit and applications. The ORM specification is based on the ORM grammar and includes information for defining the mapping between the object-oriented system and the relational system. The Object Class Definitions define the object-oriented system, and the relational database defines the relational system. The Database Exchange Unit executes in accor-